LOG3000 – INGÉNIERIE LOGICIELLE

TP4

Présenté à:
Mohammad Hamdaqa

Écrit par :
Cassy Charles (1947025)
Samy Checklat (1937812)

Remis le 2 Avril 2022

4. Questions

Lorsque possible, vous devez appuyer vos réponses avec des captures d'écran des commandes exécutées ainsi que de la sortie.

4.1 Questions d'analyse sur la section 3.1

1. Expliquez avec vos propres mots ce que fait la commande pull alpine.

```
C:\Users\cassy>docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
40e059520d19: Pull complete
Digest: sha256:f22945d45ee2eb4dd463ed5a431d9f04fcd80ca768bb1acf898d91ce51f7bf04
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
```

Figure 1: docker pull alpine

La commande pull permet l'extraction de l'image du conteneur alpine dans le registre Docker et l'enregistre dans notre système.

2. Expliquez avec vos propres mots ce qui se passe derrière l'écran lorsque vous exécutez la commande docker run.

```
C:\Users\cassy>docker run alpine ls -l
total 56
drwxr-xr-x    2 root     root          4096 Mar 28 19:26 bin
drwxr-xr-x    5 root     root           340 Mar 30 17:17 dev
drwxr-xr-x    1 root     root          4096 Mar 30 17:17 etc
drwxr-xr-x    2 root     root          4096 Mar 28 19:25 home
drwxr-xr-x    7 root     root          4096 Mar 28 19:26 lib
drwxr-xr-x    5 root     root          4096 Mar 28 19:25 media
drwxr-xr-x    2 root     root          4096 Mar 28 19:25 mnt
drwxr-xr-x    2 root     root          4096 Mar 28 19:25 opt
dr-xr-xr-x  252 root     root             0 Mar 30 17:17 proc
drwx------    2 root     root          4096 Mar 28 19:25 root
drwxr-xr-x    2 root     root          4096 Mar 28 19:25 run
drwxr-xr-x    2 root     root          4096 Mar 28 19:26 sbin
drwxr-xr-x    2 root     root          4096 Mar 28 19:25 srv
dr-xr-xr-x   11 root     root             0 Mar 30 17:17 sys
drwxrwxrwt    2 root     root          4096 Mar 28 19:25 tmp
drwxr-xr-x    7 root     root          4096 Mar 28 19:25 usr
drwxr-xr-x   12 root     root          4096 Mar 28 19:25 var
```

Figure 2: docker run alpine

La commande docker run exécute un conteneur qui a été créé auparavant ou à partir d'une image fournie. Le client Docker contacte le daemon Docker. Daemon Docker va d'abord chercher l'image dans le système local (Docker store) et vérifie si elle est disponible localement. Le daemon va donc créer le conteneur et exécute une commande dans ce conteneur de manière isolée. De cette manière, ce processus possède son propre système de fichiers, son propre réseau et sa propre hiérarchie de processus isolés, qui est indépendante de celle de la machine. Par la suite, daemon Docker envoie la sortie de la commande au client Docker.

3. Expliquez avec vos propres mots ce qui se passe derrière l'écran lorsque vous exécutez la commande docker run alpine echo "hello from alpine". S'agit-il d'une sortie de docker ou de Linux Alpine?

```
C:\Users\cassy>docker run alpine echo "hello from alpine"
hello from alpine
```

Figure 3: docker run alpine echo

Lorsqu'on exécute cette commande, on envoie une commande echo et Docker lance la commande dans le conteneur Alpine pour afficher la sortie. Il s'agit d'une sortie Docker.

4. Quelle est la différence entre une image et un conteneur?

Une image consiste en une collection de fichiers qui regroupent tous les éléments nécessaires (dépendances, code source et bibliothèques) pour créer un environnement de conteneur entièrement fonctionnel, tandis qu'un conteneur Docker est un environnement d'exécution virtualisé qui offre des capacités d'isolation pour séparer l'exécution des applications du système sous-jacent.

Les images peuvent exister sans conteneur, alors qu'un conteneur a besoin d'exécuter une image pour exister. Par conséquent, les conteneurs sont dépendants des images et les utilisent pour construire un environnement d'exécution et exécuter une application.

5. Quels sont les avantages d'un conteneur par rapport à une machine virtuelle?

Les conteneurs virtualisent le système d'exploitation de sorte que chaque conteneur individuel ne contient que l'application, ses bibliothèques et ses dépendances. Les conteneurs sont petits, rapides et portables car, contrairement à une machine virtuelle, les conteneurs n'ont pas besoin d'inclure un système d'exploitation invité dans chaque instance et peuvent, au contraire, simplement exploiter les fonctionnalités et les ressources du système d'exploitation hôte. La virtualisation dans un conteneur est gérée plus facilement. Ils prennent aussi moins d'espace que les machines virtuelles.

4.2 Questions d'analyse sur la section 3.2 6.

Expliquez avec vos propres mots chaque paramètre utilisé à l'étape 4 de la section 3.2.

```
C:\Users\cassy>docker run --name static-site -e AUTHOR="Cassy" -d -P dockersamples/static-site
8083d90c23afcc799915e1bddac24405307bdaff6ce7df231707807cb956421d
```

Figure 4: docker run name static-site

docker run --name static-site -e AUTHOR="" -d - P dockersamples/static-site

--name: Il est possible d'attribuer des noms nous-mêmes qu'on peut mémoriser à des conteneurs Docker quand ils sont exécutés

static-site : Le nom donné au conteneur docker

-e: Il est aussi possible de définir une variable d'environnement dans le conteneur.

AUTHOR= « » : Ajoute l'étiquette de l'auteur

-d : Ce paramètre permet de démarrer le conteneur en mode détaché

-p : Ce paramètre permet de publier tous les ports sur l'interface de l'hôte. Il permet aussi de mapper explicitement un port unique ou multiples ports.

Dockersamples/static-site : l'image du conteneur

4.3 Questions d'analyse sur la section 3.3

7. Expliquez la sortie de la commande docker images. Comment obtenir une version spécifique d'une image?

```
C:\Users\cassy>docker images
REPOSITORY              TAG       IMAGE ID       CREATED        SIZE
alpine                  latest    76c8fb57b6fc   41 hours ago   5.57MB
docker/getting-started  latest    bd9a9f733898   6 weeks ago    28.8MB
hello-world             latest    feb5d9fea6a5   6 months ago   13.3kB
```

Figure 5: docker images

La sortie de la commande docker images affiche toutes les images de premier niveau, leur répertoire et leurs balises(tag), leur date de création ainsi que leur taille.

Pour obtenir une version spécifique d'une image, on peut le faire directement à partir de la commande docker images image_name : version.

Exemple :

```
C:\Users\cassy>docker images
REPOSITORY                  TAG       IMAGE ID       CREATED        SIZE
ccassyie/dockerapp          latest    70a1d79a780b   5 hours ago    142MB
<none>                      <none>    856e35d6a509   5 hours ago    142MB
<none>                      <none>    ac8067f52e77   5 hours ago    142MB
<none>                      <none>    40d8c3fe2747   5 hours ago    142MB
ccassyie/webserver          latest    ea1a5f7e2025   5 hours ago    142MB
<none>                      <none>    395871219666   5 hours ago    142MB
<none>                      <none>    b6d2c454b453   5 hours ago    142MB
<none>                      <none>    095bece2a43b   6 hours ago    23.4MB
<none>                      <none>    72a69d62be0b   6 hours ago    142MB
ccassyie/myfirstapp         latest    6c15f012419a   6 hours ago    56.8MB
<none>                      <none>    7301e55b70af   6 hours ago    56.8MB
alpine                      latest    76c8fb57b6fc   2 days ago     5.57MB
docker/getting-started      latest    bd9a9f733898   6 weeks ago    28.8MB
hello-world                 latest    feb5d9fea6a5   6 months ago   13.3kB
dockersamples/static-site   latest    f589ccde7957   6 years ago    191MB

C:\Users\cassy>docker images hello-world:latest
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    feb5d9fea6a5   6 months ago   13.3kB
```

Figure 6: docker images specific_image

8. Expliquez avec vos propres mots la différence entre base images et child images.

Une base images est une image utilisée pour créer toutes les images d'un conteneur. Un utilisateur peut créer sa propre image de base dans un docker à partir de rien et l'adapter à ses besoins.

Un child images est une image qui a déjà une image parent dans son Dockerfile

9. Expliquez avec vos propres mots la différence entre official images et user images.

Official images sont un ensemble de répertoires de Docker qui sont hébergés sur Docker Hub. Elles fournissent des solutions pour les moteurs d'exécution des langues de programmation similaire au PAAS. De plus, elles illustrent les meilleures pratiques de dockerfile et s'assurent que la sécurité des images est appliquée en temps voulu.

User images sont des images provenant des utilisateurs. D'abord, il faut lancer un docker run pour créer un conteneur à partir d'une image fournie et la démarrer avec la commande correspondante.

10. Expliquez avec vos propres mots ce qu'est un Dockerfile.

Un Dockerfile est un document texte similaire à un script qui contient toutes les commandes à exécuter pour créer une image Docker. On peut l'appeler "instructions d'installation d'une image sous forme de code", mais uniquement pour Docker et de manière beaucoup plus simple.

11. Expliquez chaque ligne du Dockerfile créé à l'étape 5 de la section 3.3.

FROM alpine:3.5 : spécifie l'image parentale à partir de laquelle on construit qui est alpine 3.5.

RUN apk add --update py2-pip : Roule les mises à jour des dépendances pour python

COPY requirements.txt /usr/src/app/ : COPY permet de copier un fichier d'une source vers dossier de destination qui est dans le système de fichiers. Ici, on copie le fichier requirements.txt vers /usr/src/app/.

RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt : Cette ligne installe les éléments listés dans le requirements.txt pour que l'application puisse les utiliser.

COPY app.py /usr/src/app/ : Ici le fichier app.py est copié vers le dossier /usr/src/app/ qui se trouve dans le système de fichiers.

COPY templates/index.html /usr/src/app/templates/ : Ici le fichier index.html est copié vers le dossier */usr/src/app/templates/* qui se trouve dans le système de fichiers.

EXPOSE 5000 : Cette ligne informe que le conteneur écoute le port réseau spécifié quand l'exécution a été lancée. Elle ne publie pas réellement le port mais permet de clarifier que le conteneur écoute sur le port 5000.

CMD ["python", "/usr/src/app/app.py"] : Cette ligne permet de définir des valeurs par défaut supplémentaires qui peuvent être modifiées.

12. Expliquez ce qui se passe lorsque vous exécutez la commande docker build -t /flask-app.

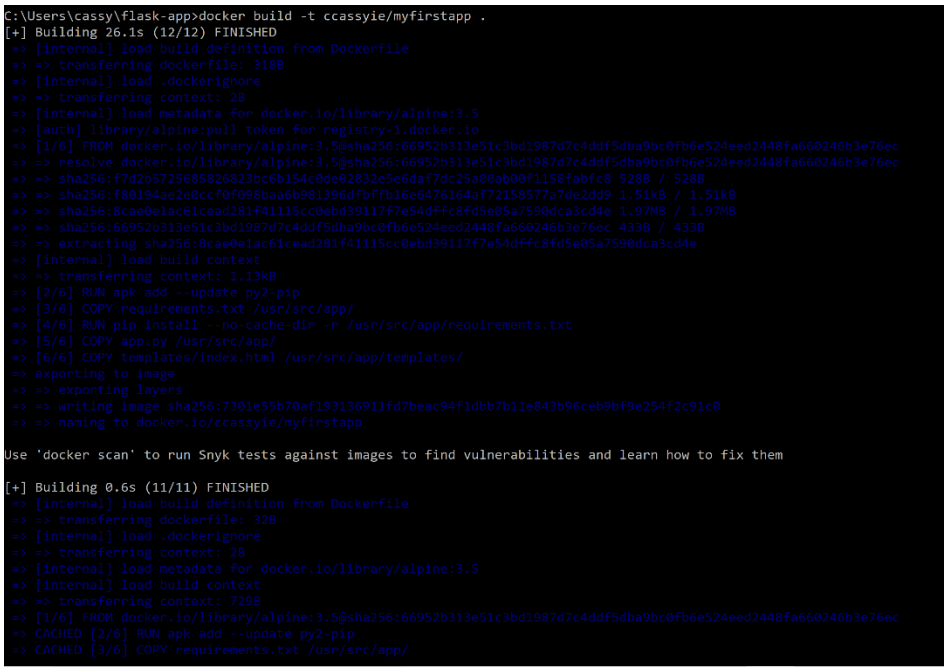

Figure 7: docker build flask-app

Cette commande permet de construire l'image fournie à partir des instructions du Dockerfile. À cette image est ajoutée une balise grâce au -t dans le répertoire de l'utilisateur après sa construction.

4.4 Questions d'analyse sur la section 3.4

13. Décrivez chaque ligne du Dockerfile que vous avez créé. Ajoutez des captures d'écran et la sortie des commandes pour appuyer votre explication. Ajoutez également une image de votre navigateur montrant l'URL et le site Web en cours d'exécution.

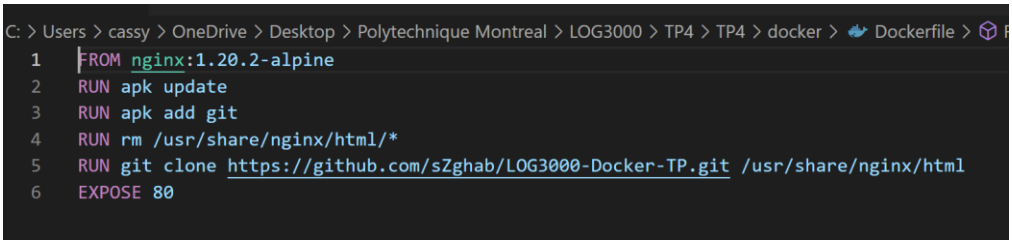

Figure 8: Dockerfile for nginx

FROM nginx:1.20.2-alpine : Ici, on spécifie l'image de base de Nginx sur laquelle on va construire notre image pour ajouter nos éléments

RUN apk update : Ici, on roule les dernières mises à jour des paquets du système de fichiers

RUN apk add git: Ici, ajoute git à notre système de fichiers

*RUN rm /usr/share/nginx/html/** : Ici on enlève les fichiers compris dans /usr/share/nginx/html/*

RUN git clone https://github.com/sZghab/LOG3000-Docker-TP.git /usr/share/nginx/html : Ici, on copie notre fichier à partir du site web fourni et on l'ajoute dans notre dossier /usr/share/nginx/html.

EXPOSE 80 : Ici, le conteneur écoute sur le port réseau 80 lors de l'exécution

Il n'était pas nécessaire d'avoir des RUN, car aucune mise à jour ni de dépendances n'est à installer, vu qu'on travaille avec un fichier HTML et un fichier .js.

Par la suite, on lance un docker build pour construire l'image faite à partir de notre Dockerfile.

```
C:\Users\cassy\OneDrive\Desktop\Polytechnique Montreal\LOG3000\TP4\TP4\docker>docker build -t ccassyie/dockerapp .
[+] Building 9.8s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 224B                                                                    0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/library/nginx:1.20.2-alpine                                  1.4s
 => [auth] library/nginx:pull token for registry-1.docker.io                                            0.0s
 => [1/5] FROM docker.io/library/nginx:1.20.2-alpine@sha256:ff557e536e5c697c5a28db13ab81bdf9b0c6a20161aa0a46419b9  3.2s
 => => resolve docker.io/library/nginx:1.20.2-alpine@sha256:ff557e536e5c697c5a28db13ab81bdf9b0c6a20161aa0a46419b9  0.0s
 => => sha256:1865a131612a5b8407d596a035b6ce1fa53c94f2f1b175c52d110565192d2f0d 1.57kB / 1.57kB          0.0s
 => => sha256:cfe7b895dacabd191c9a48c9818e0e4b41d7776249705cc33365ee60a025c33d 8.88kB / 8.88kB          0.0s
 => => sha256:cfab2db722092277479ddd659049695fa8081d2a03c31b01d388cc21802de8b4 2.82MB / 2.82MB          0.7s
 => => sha256:d4befe7855e09ae6defb44d9cc2aac6a2483072cfba453a5a8098ddcf31bf25c 7.23MB / 7.23MB          1.9s
 => => sha256:9fa808a522b662e4dba75d33a64088473ee11bd5d0dbf9ac534ba160bc27686c 601B / 601B              0.7s
 => => sha256:ff557e536e5c697c5a28db13ab81bdf9b0c6a20161aa0a46419b9b251872c7df 1.65kB / 1.65kB          0.0s
 => => extracting sha256:cfab2db722092277479ddd659049695fa8081d2a03c31b01d388cc21802de8b4               0.3s
 => => sha256:bc33f0e186629973d764e9fa1bb14d35d13cd518592441df9a60efc0a67bf30a 894B / 894B              0.8s
 => => sha256:54124d323c22c05ff35c5036de785a025b59c520ca70bdde5ab85e91400d9d7d 665B / 665B              0.8s
 => => sha256:afbe9b76fe538a595b1b6a9e1b2fdabcc54e0323e2ca6698f96eee9d4176e6f4 1.39kB / 1.39kB          1.0s
 => => extracting sha256:d4befe7855e09ae6defb44d9cc2aac6a2483072cfba453a5a8098ddcf31bf25c               0.6s
 => => extracting sha256:9fa808a522b662e4dba75d33a64088473ee11bd5d0dbf9ac534ba160bc27686c               0.0s
 => => extracting sha256:bc33f0e186629973d764e9fa1bb14d35d13cd518592441df9a60efc0a67bf30a               0.0s
 => => extracting sha256:54124d323c22c05ff35c5036de785a025b59c520ca70bdde5ab85e91400d9d7d               0.0s
 => => extracting sha256:afbe9b76fe538a595b1b6a9e1b2fdabcc54e0323e2ca6698f96eee9d4176e6f4               0.0s
 => [2/5] RUN apk update                                                                                1.5s
 => [3/5] RUN apk add git                                                                               2.0s
 => [4/5] RUN rm /usr/share/nginx/html/*                                                                0.6s
 => [5/5] RUN git clone https://github.com/sZghab/LOG3000-Docker-TP.git /usr/share/nginx/html          0.9s
 => exporting to image                                                                                  0.1s
 => => exporting layers                                                                                 0.1s
 => => writing image sha256:27b83dfedbe42663ad97f66f0eca735bceeceac7f9ada046f0ead5c5eef99f30            0.0s
 => => naming to docker.io/ccassyie/dockerapp                                                           0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

Figure 9: docker build dockerapp from nginx

Et pour afficher l'image sur la page web, on fait un docker run avec le port 8080.

```
C:\Users\cassy\OneDrive\Desktop\Polytechnique Montreal\LOG3000\TP4\TP4\docker>docker run -it --rm -d -p 8080:80 --name d
ockerapp ccassyie/dockerapp
1c48bad74f80f0718f0252ce735a720bb5f4c46239b387d835b4556854715e5e
```

Figure 10: docker run

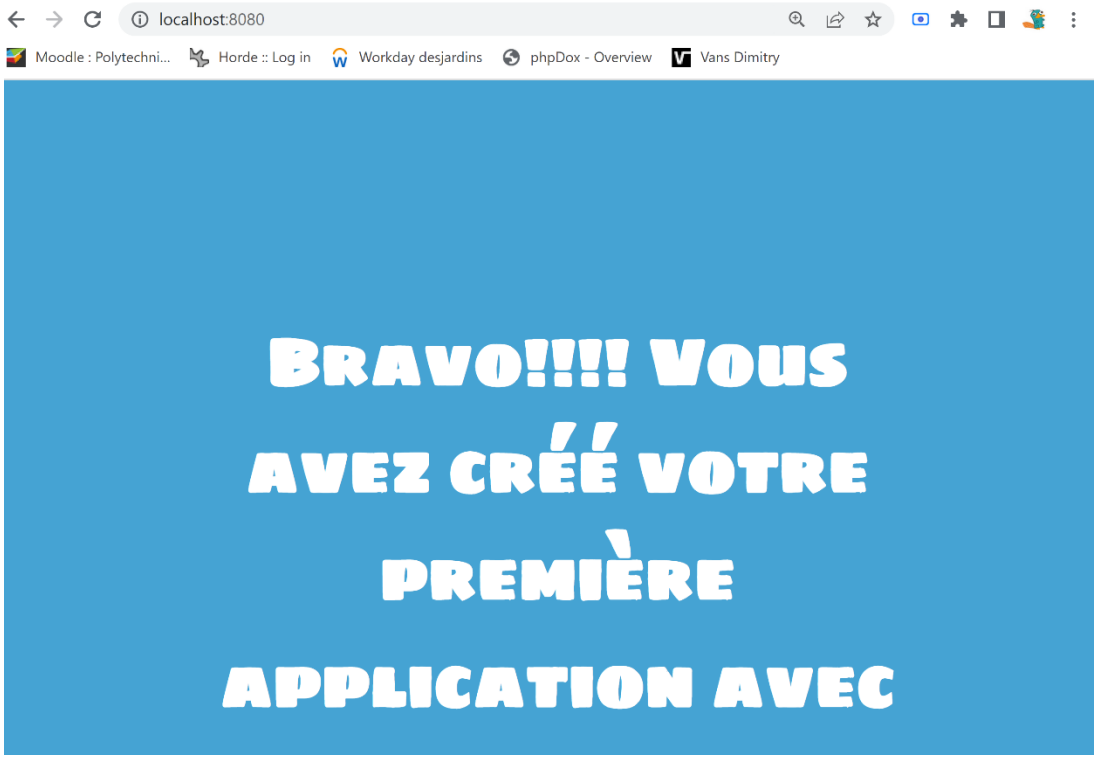

Figure 11: site web nginx

4.5 Question de rétroaction

14. Nous travaillons à l'amélioration continue des travaux pratiques de LOG3000. Cette question peut être répondue très brièvement.

15. Combien de temps avez-vous passé au travail pratique, en heures-personnes, en sachant que deux personnes travaillant pendant trois heures correspondent à six heurespersonnes. Est-ce que l'effort demandé pour ce laboratoire est adéquat ?

Nous avons passé 5 heures-personnes sur la réalisation de ce laboratoire. Oui, l'effort est adéquat.

16. Quelles difficultés avez-vous rencontré lors de ce laboratoire?

Les difficultés rencontrées se sont présentées lors de la création du Dockerfile, vu que c'était notre première utilisation de Docker.